Université d'Ottawa
Faculté de génie

École de science
d'informatique
et de génie électrique

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

School of Electrical
Engineering
and Computer Science

Assignment 2
CSI2120 Programming Paradigms
Winter 2017

Due on February 27th, 2017 before 11:00 pm in Virtual Campus

[5 marks in total]

Question 1. **[1.5 marks]**

Write a generator which produces all positive numbers X divisible by a number D below an upper bound U.

a) Design a predicate the predicate `divisible( D, U, X ).`

```
?- divisible( 4, 15, X ).
X = 4 ;
X = 8 ;
X = 12
```

b) Design a predicate that collects all divisors (the solutions from part a) into a list.

```
?- divisibleAll( 4, 15, L ).
L = [ 4, 8, 12]
```

Question 2. **[3.5 marks]**

A wizard has 16 different hats with four different colours and four different letters. The four colours of the hats are red, green, yellow and blue, the four different letters are w,x,y and z. No two hats are the same. The wizard stores the hats in a drawer with a four by four grid. Because of wizardry, the hats need to be stored in such a way that in each column and each row of the drawer, no two hats have the same colour or the same letter. You are asked to help the wizard to find a solution to put the wizard's hat in the drawer.

The wizard's drawer:

a) Design a predicate `generateHats/1` that generates a list of all possible hats.

```
?-  generateHats(L).
L = [ (blue, z), (blue, y), (blue, x), (blue, w), (yellow, z),
(yellow, y), (yellow, x), (yellow, w), (..., ...)|...].
```

b) Write a predicate `validRow/1` that is true if a row is valid, i.e., if no two hats have the same color or the same letter.

```
?- validRow([ (1, 1, red, w), (1, 2, green, x), (1, 3, yellow, y),
(1, 4, blue, z)]).
true.
```

Note that the row here is 1 and the columns are numbered with 1,2,3,4. Your predicate should check the colors and letter for all entries with the same row number in the list.

c) Write a predicate `generateRow/3` to generate all valid rows by picking hats from a list.

```
?- generateHats(H), generateRow(H,1,R).
H = [ (blue, z), (blue, y), (blue, x), (blue, w), (yellow, z),
(yellow, y), (yellow, x), (yellow, w), (..., ...)|...],
R = [ (1, 1, redd, w), (1, 2, green, x), (1, 3, yellow, y), (1, 4,
blue, z)] ;
H = [ (blue, z), (blue, y), (blue, x), (blue, w), (yellow, z),
(yellow, y), (yellow, x), (yellow, w), (..., ...)|...],
R = [ (1, 1, red, x), (1, 2, green, w), (1, 3, yellow, y), (1, 4,
blue, z)]
```

Note that the row here the input argument 1 determines the row number in the result.

d) Write a predicate uniqueRows/1 that tests if all rows in a list are compatible, i.e., each column has to contain hats that differ in colour and letter.

```
?- uniqueRows([ (1, 1, red, w), (1, 2, green, x), (1, 3, yellow,
y), (1, 4, blue, z), (2, 1, green, x), (2, 2, red, w), (2, 3, blue,
z), (2, 4, yellow, y)]).
true.
```

e) Write a predicate `wizardry/1` that produces a solution to the problem of placing hats in a drawer. Find the solution by generating a valid row and removing the hats from the set of available hats, then generating a second valid row, checking that the two rows are compatible and then again removing the corresponding hats from the list of hats. Keep going until you find 4 compatible rows and you have placed all 16 hats. Note that your predicate should find all solutions.

```
?- wizardry(L).
L = [ (1, 1, red, w), (1, 2, green, x), (1, 3, yellow, y), (1, 4,
blue, z), (2, 1, green, y), (2, 2, red, z), (2, 3, blue, w), (2, 4,
yellow, x), (3, 1, yellow, z), (3, 2, blue, y), (3, 3, red, x), (3,
4, green, w), (4, 1, blue, x), (4, 2, yellow, w), (4, 3, green, z),
(4, 4, red, y)] ;
L = [ (1, 1, red, w), (1, 2, green, x), (1, 3, yellow, y), (1, 4,
blue, z), (2, 1, green, y), (2, 2, red, z), (2, 3, blue, w), (2, 4,
yellow, x), (3, 1, blue, x), (3, 2, yellow, w), (3, 3, green, z),
(3, 4, red, y), (4, 1, yellow, z), (4, 2, blue, y), (4, 3, red, x),
(4, 4, green, w)]
```

f) Write a predicate drawer which prints the drawer content in matrix format.

```
?- drawer([ (1, 1, red, w), (1, 2, green, x), (1, 3, yellow, y),
(1, 4, blue, z), (2, 1, green, y), (2, 2, red, z), (2, 3, blue, w),
(2, 4, yellow, x), (3, 1, yellow, z), (3, 2, blue, y), (3, 3, red,
x), (3, 4, green, w), (4, 1, blue, x), (4, 2, yellow, w), (4, 3,
green, z), (4, 4, red, y)]).
1,1,red,w    1,2,green,x    1,3,yellow,y    1,4,blue,z
2,1,green,y    2,2,red,z    2,3,blue,w    2,4,yellow,x
3,1,yellow,z    3,2,blue,y    3,3,red,x    3,4,green,w
4,1,blue,x    4,2,yellow,w    4,3,green,z    4,4,red,y
true.
```